TIPURI DE DATE ABSTRACTE

Avantajele tipurilor de date abstracte:

- Permit o abstractizare prin care obiectele se clasifică nu numai conform structurii, ci şi conform operațiilor corespunzătoare.

- Permit verificarea riguroasă, la compilare a utilizării corecte.

- Reprezintă un suport de programare excelent pentru implementarea metodologiei mascării informației.

1. Descrierea datelor abstracte.

- Descrierea unui tip abstract de stivă (într-un limbaj imaginar):

```
type stiva(nr_max:integer)=class
        operations pop, push, top;
        var tab_st:array[1..nr_max] of integer;
            ind:integer;

        function pop():integer;
          begin
            if not empty() then
                ind:=ind-1;
                return tab_st[ind];
            else
                exceptie;
            end if;
          end;
  procedure push(x:integer);
 begin
    if not overflow() then
        tab_st[ind]:=x;
        ind:=ind+1;
    else
        exceptie;
    end if;
 end;
function top():integer;
begin
    if not empty() then
        return tab_st[ind-1];
    else
        exceptie;
    end if;
end;
```

```
function empty():boolean;
begin
        return ind=1;
end;
function overflow():boolean;
begin
        return ind > nr_max;
end;
begin
        ind:=1;
end;
```

1. Descrierea datelor abstracte se oferă printr-o construcție de tip.

- Declararea variabilelor de tip *stiva (*specificând dimensiunea totală a stivei*)*:

```
var st1, st2 : stiva(100);
    st3 : stiva(55);
```

- Specificarea operațiilor care se pot executa asupra variabilelor declarate.

```
st1.push(10);
--------------
st2.push(k);
-------------
i:=st1.top();
------------
j:=st2.pop();
```

2. **Descrierea datelor abstracte – printr-un modul care nu reprezintă un constructor de tip**.

```
module st;
    export stiva,pop,push,top,init;
    type stiva(nr_max:integer)=record
              Tab_st:array[1..nr_max] of integer;
              ind:integer; end record;
    function pop(var s:stiva):integer;
    begin
      if not empty(s) then
            s.ind:=s.ind-1;
            return s.tab_st[s.ind];
      else exceptie;
      end if;
    end;
    procedure push(var s:stiva;x:integer);
    begin
       if not overflow(s) then
             s.tab_st[s.ind]:=x;  s.ind:=s.ind+1;
       else exceptie;
       end if;
    end;
    function top(s:stiva):integer;
    begin
       if not empty(s) then
             return s.tab_st[s.ind-1]
       else  exceptie;  end if;
    end;
    procedure init(var s:stiva);
    begin   s.ind:=1;   end;
    function  empty(var s:stiva):boolean;
    begin   return s.ind=1;  end;
    function  overflow(var s:stiva):boolean;
    begin   return s.ind>s.nr_max;  end;
```

- descriere de variabile de tipul *stiva:*

```
var st1, st2:st.stiva(100);
    st3:st.stiva(50);
```

- operații asupra acestor obiecte:

```
st.init(st1);
st.init(st2);
st.init(st3);
----------
st.push(st1,10);
----------
st.push(st2,k);
---------
i:=st.top(st1);
---------
j:=st.pop(st2);
```

- programatorul nu are acces la câmpurile variabilei de tip *stiva*.

- Programatorul poate să declare variabile de tipul *stiva* şi poate să-l transmită ca parametru atunci când apelează operațiile exportate din modulul *st*.

3. Tipuri generice sau polimorfe.

```
type stiva(type tip_el, nr_max:integer)=class
   operation pop, push, top;
   var var_st:array[1..nrmax]of tip_real;
        Ind:integer;
   function pop():tip_el;
        begin
        ------
        end;
   procedure push(x:tip_el);
        begin
       -------
        end;
   function top():tip_el;
        begin
        ------
        end;
function empty():boolean;
        begin   ------   end;
function overflow():boolean;
        begin   ------   end;
begin
---------
end;
```

- utilizarea tipului *stiva.*

```
type t=--------
var  st1 : stiva(integer,100);
     st2 : stiva(real,150);
     st3 : stiva(t,50);
--------------
st1.push(10);
st2.push(1.5);
```

Polimorfism cu module care nu sunt constructori de tip.

```
module st( type tip_el);
   export stiva, pop, push, top, init;
   type stiva( nr_max:integer) = record
                     Tab_st:array[1..nr_max] of tip_el;
                     Ind:integer;
           end record;
   function  pop(var s:stiva) : tip_el ;
       begin
       ------
       end;
   procedure  push(var s:stiva; x:tip_el) ;
       begin
       ------
       end;
   function  top(var s:stiva) : tip_el;
       begin
       ------
       end;
   procedure  init(var s:stiva) ;
       begin
       ------
       end;
   function empty(var s:stiva) : boolean ;
       begin
       ------
       end;
   function overflow(var s:stiva) : boolean;
       begin
       ------
       end;
begin
-------
end;
```

- Modulul *st* poate fi instanţiat pentru orice tip de element pentru care apoi se doreşte declararea unui obiect stivă:

```
type t = ----------;
module st_int = st(integer);
module st_real = st(real);
module st_t = st(t);
```

Declararea obiectelor stivă:

```
Var  st1 : st_int.stiva(100);
     st2: st_real.stiva(150);
     st3: st_t.stiva(50);
---------------------------------
     st_int.push(st1,10);
---------------------------------
     st_real.push(st2,1.5);
---------------------------------
```